World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:10, No:4, 2016

Multi-Objective Random Drift Particle Swarm Optimization Algorithm Based on RDPSO and Crowding Distance Sorting

Yiqiong Yuan, Jun Sun, Dongmei Zhou, Jianan Sun

Abstract—In this paper, we presented a Multi-Objective Random Drift Particle Swarm Optimization algorithm (MORDPSO-CD) based on RDPSO and crowding distance sorting to improve the convergence and distribution with less computation cost. MORDPSO-CD makes the most of RDPSO to approach the true Pareto optimal solutions fast. We adopt the crowding distance sorting technique to update and maintain the archived optimal solutions. Introducing the crowding distance technique into MORDPSO can make the leader particles find the true Pareto solution ultimately. The simulation results reveal that the proposed algorithm has better convergence and distribution.

Keywords—Multi-objective optimization, random drift particle swarm optimization, crowding distance, Pareto optimal solution.

I. Introduction

OPTIMIZATION is considered as one of the most important problems in the fields of mathematics and science. In the real world, we usually need to optimize multiple objectives at the same time, and these objectives are often contradictory. There is normally no unique optimal solution which make every objective of the optimization problem optimal. Therefore, coordinating and compromising the optimization objectives are critical to solve the multi-objective optimization problem. The multi-objective evolutionary algorithm is one of the effective methods to solve this problem [1]. On account of the good convergence, simple calculation and less parameters, multi-objective particle swarm optimization algorithm has received extensive attention and research recently.

Since Coello and Lechuga [2] formally put forward multi-objective particle swarm optimization (MOPSO) in 2002, people have learned some methods from multi-objective evolutionary algorithms to improve the multi-objective particle swarm optimization (MOPSO) algorithm, emerging many improved algorithms, such as adopting crowding distance sorting, clustering technique [3], niching technique [4] and adaptive grid [5], [6], so on. All of these improvements aim at solving two key problems: 1) How to find the true Pareto solution set; 2) How to maintain the diversity of Pareto solution.

Classical Particle Swarm Optimization (PSO) algorithm exists the defect of low local search accuracy. The Quantum-behaved Particle Swarm Optimization (QPSO) [7] algorithm has stronger search ability, faster convergence and less parameters, which can improve weak ability and low local searching precision in PSO. But RDPSO algorithm has better performance compared with QPSO. So we introduce it into solving the multi-objective optimization problem. But when we use it to solve the multi-objective optimization problem, fast convergence means that the algorithm is easy to premature convergence and lose the diversity of solutions. Therefore, we need to introduce crowding distance sorting to maintain the diversity of Pareto solutions and find the true Pareto optimal solution finally. Crowding distance sorting can fully reflect the density information in spite of high computational complexity, and it is conducive to cut the particles with high redundancy and reserve the particles which have good global search capability, so it is able to maintain the diversity of solutions, this property can be proved from [8].

In this paper, we presented a multi-objective optimization algorithm–Multi-Objective Random Drift Particle Swarm Optimization algorithm based on RDPSO and crowding distance sorting. Meanwhile, the simulation results reveal that the proposed algorithm has better convergence and distribution.

II. Multi-Objective Optimization Problem Description

Here are the concepts [9] that are commonly used in the multi-objective optimization problem.

A. Definition 1: Multi-Objective Optimization Problem

Assume that we solve the multi-objective minimization problem (a maximization problem can reverse into the minimization problem through taking opposite or reciprocal). The mathematical model of the multi-objective optimization problem can be described as:

$$\begin{cases} min\ y = F(x) = (f_1(x), f_2(x), ..., f_m(x)). \\ \quad s.t. g_i(x) \le 0, i = 1, 2, ...h; \\ \quad x \in X \in R^n,\ y \in Y \in R^m, \end{cases} \tag{1}$$

where, x is the decision vector, y is the objective vector; X is considered as the decision space which is formed by decision vector x, Y is considered as the target space which is formed by target vector y; f is the optimization function which maps x to target vector space.

Yiqiong Yuan is with the School of IoT Engineering, Jiangnan University, Wuxi, Jiangsu, 214122 China (phone: +86 18206181656; e-mail: yyqmsunshine@163.com).

Jun Sun is with the Key Laboratory of Advanced Process Control for Light Industry(Ministry of Education), Jiangnan University, Wuxi, Jiangsu, 214122 China (phone: +86 13665180007).

B. Definition 2: Pareto Dominate

Set X_f as the feasible solution set of multi-objective optimization problem and $F(x) = (f_1(x), f_2(x), ..., f_m(x))$ as the target vector. Suppose that $x_k \in X_f, x_l \in X_f$, x_k Pareto dominate x_l (written for$x_k \prec x_l$) if and only if

$$\begin{cases} \forall i \in 1, 2, ..., m : f_i(x_k) \leq f_i(x_l) \\ \exists j \in 1, 2, ..., m : f_j(x_k) \prec f_j(x_l) \end{cases} \quad (2)$$

C. Definition 3: Pareto Optimal Front

Pareto optimal front is composed of objective vectors set which are in correspond with Pareto optimal solutions for a multi-objective optimization problem.

Different from the single problem, the optimal solution for the multi-objective optimization problem does not exist, which only has Pareto optimal solution. In general, though there are many Pareto optimal solutions in the multi-objective optimization problem, the Pareto optimal solution for the multi-objective optimization problem is proved to be only an acceptable "satisfied solution"; improving the performance of any one objective will inevitably lead to the depressed performance of others.

III. Random Drift Particle Swarm Optimization Algorithm

RDPSO is one of the swarm intelligence methods inspired by the free electron model of metal conductors in an external electric field [10]. It adopts a novel set of evolution equations which is able to enhance the ability of global search. Through trajectory analyzing, we can see that the convergence of the PSO algorithm can be achieved by each particle converging to its local attractor, we define $P_{i,n} = (P^1_{i,n}, P^2_{i,n}, .., P^n_{i,n})$ as the coordinates, and

$$P^j_{i,n} = \frac{c_1 r^j_{i,n} P^j_{i,n} + c_2 R^j_{i,n} G^j_n}{c_1 r^j_{i,n} + c_2 R^j_{i,n}}, 1 \leq j \leq N, \quad (3)$$

where, $r^j_{i,n}$ and $R^j_{i,n}$ are the random numbers on (0, 1). The acceleration coefficients c_1 and c_2 generally are set to 2. Thus, (3) can be transformed into (4).

$$P^j_{i,n} = \varphi^j_{i,n} P^j_{i,n} + (1 - \varphi^j_{i,n}) G^j_n, \quad (4)$$

where,

$$\varphi^j_{i,n} = \frac{c_1 r^j_{i,n}}{c_1 r^j_{i,n} + c_2 R^j_{i,n}}.$$

The particles directional movement toward is similar to the drift motion of an electron in a metal conductor placed in an external electric field [11]. Based on this fact, we suppose that the behavior of particle in RDPSO is resemble to the movement of an electron in a metal conductor in an external electric field. Consequently, there are two parts as to the movement of particle towards $P^j_{i,n}$: the thermal motion and the drift motion.

Thus, the particles velocity can be represented by $V^j_{i,n} = VR^j_{i,n} + VD^j_{i,n}$, where $VR^j_{i,n}$ and $VD^j_{i,n}$ respectively delegate the velocities of the thermal motion and the drift motion towards $P^j_{i,n}$. RDPSO algorithm only needs to modify the particle's velocity updating equation, which reduces the complexity of the algorithm and computational cost. According to the following expression, we update $VR^j_{i,n}$ and $VD^j_{i,n}$:

$$VR^j_{i,n+1} = \sigma^j_{i,n} \lambda^j_{i,n+1}. \quad (5)$$

$$VD^j_{i,n} = \beta(P^j_{i,n} - X^j_{i,n}). \quad (6)$$

$\sigma^j_{i,n}$ in (5) is the standard deviation of a Gaussian distribution, whose value is determined adaptively by

$$\sigma^j_{i,n} = \alpha |C^j_n - X^j_{i,n}|, \quad (7)$$

where, $C^j_n = (1/M) \sum_{i=1}^{M} P^j_{i,n}, (1 \leq j \leq N)$ is considered as the best mean position. α and β are two positive real numbers called the thermal coefficient and the drift coefficient. Obviously, the velocities of the thermal motion $VR^j_{i,n}$ obey the Maxwell velocity distribution law. From (6), we can see that the velocities of the drift motion $VD^j_{i,n}$ make particle move towards $P^j_{i,n}$ in each iteration. The drift coefficient β plays a decisive role, the particles local searching capability is better when $1 \leq \beta \leq 2$. Therefore, we adopt the following evolution equations for RDPSO.

$$V^j_{i,n+1} = \alpha |C^j_n - X^j_{i,n}| \lambda^j_{i,n+1} + \beta(P^j_{i,n} - X^j_{i,n}. \quad (8)$$

$$X^j_{i,n+1} = X^j_{i,n} + V^j_{i,n+1} \quad (9)$$

IV. A Multi-Objective Random Drift Particle Swarm Optimization Algorithm Based on Crowding Distance Sorting (RDPSO-CD)

A. The Basic Theory of Crowding Distance

Crowding distance [12] is used to measure the proximity between a point and its neighbors. Without loss of generality, our derivation will be highlighted with two objectives, but the results can be easily extended to multiple objectives. The crowding distance of the solution can be obtained by:

$$i_{distance} = (1/2)(i^{f_1}_{dist} + i^{f_2}_{dist}), \quad (10)$$

where, $i^{f_1}_{dist}$ and $i^{f_2}_{dist}$ are the adjacent edge in this rectangle as shown in Fig. 1. Crowding distance can be used to evaluate size of the cuboid, contains only the point without any other points.

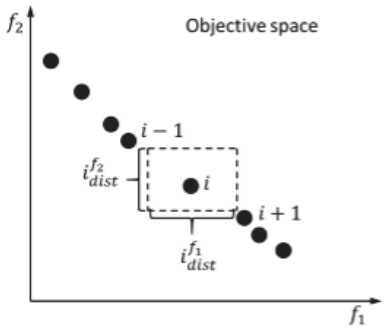

Fig. 1 Diagram of calculating the i^th point crowding distance

The process of calculating the crowding distance of each point in solution set T, is as:

Step 1.Count the number of solutions l and initialize crowding distance for each point in set $TT^i_{distance} = 0, i = 1, 2, ..., l$, and set the objective function counter $j = 1$.

Step 2.Sort the solution according to the j^{th} objective function value $f_j(T)$, the new solution set written as TS.

Step 3.In set TS, define the first and the last as positive infinity: $TS^1_{distance} = TS^l_{distance} = \infty$.

Step 4.Calculate crowding distance of the rest solution $TS^i_{distance}(2 \leq i \leq l-1)$ according to (11):

$$TS^1_{distance} = TS^i_{distance} + (TS^{i+1}_j - TS^{i-1}_j), \quad (11)$$

where, TS^{i+1}_j is the $(i+1)^th$ solution corresponding j^th objective function value.

Step 5.$j = j + 1$, return to step 2.

In order to ensure uniformity of solutions, we need to lay off redundant solutions. For further optimization, we are inclined to choose the solutions with larger crowding distances. In the light of the above process, we know that the marginal points are apt to be selected. The higher average crowding distance, the greater diversity of population.

B. External File Updating and Maintenance Strategy Based on Crowding Distance

External file (archive) first appeared in the SPEA algorithm. It is utilized to store all the current optimal solution which do not dominate each other in the operation. If the new solution dominates the one or some of the solutions, or augmenting new solution leads to the external memory exceeding the maximum capacity, we need to update and maintain the external memory, operation process is listed:

Step 1.If the new solution dominates the one or some of the solutions, we should delete the solutions which are dominated in the external memory and augment new solution. If the new solution do not dominate any others, then we just add the new solution.

Step 2.Judge whether the current external memory exceeds the maximum capacity. If it exceeds the allowed capacity, we need to perform cutting operation; and if not, updating and maintaining are aborted.

Step 3.Calculate the crowding distance of all solutions in the external memory and sort those in descending order.

Step 4.According to the order in step 3, if there are two or more in last, then we should randomly choose one of them and shift it out. Return to step 2.

C. The Selection Mechanism of Global Best Particle

The global best particle (leader particle) mainly guide particles globally searching, the choice of the leader particle is directly related to whether the algorithm can find Pareto solution set. With the propose of the diversity and distribution of solutions, leader particle of each solution can be determined according to crowding distance of the existing optimal solution in the external memory. The solution with higher crowding distance distribute more uniformly. Hence, after calculating crowding distances of all leader particles in external memory, we sort all of them in descending order. Then, taking the top ten percent of the particles as candidate leader particles of the current solution. Finally, each solution randomly chooses its own leader from these candidate particles.

D. MORDPSO-CD Algorithm Flow

According to the above algorithm thought and principle, the main pseudo-code MORDPSO-CD algorithms is as:

Algorithm 1 The MORDPSO-CD algorithm

Begin:
Pre-assign M, N, V_{max}, α and β
Initialize the current positions and velocities of all particles randomly;
Set the current positions as the personal best positions of each particle;
Evaluate the objective values of the personal best positions;
Initialize the external memory using the Pareto-dominated rules;
Set $n = 0$
while not(termination condition) **do**
$n = n + 1$
for $i = 1 \rightarrow M$ **do**
Assign $G_{i,n}$ for each particle according to Section IV.*C*
Update $X_{i,n}$ and $V_{i,n}$ according to (8) and (9);
Evaluate the objective values of the particles;
Update $P_{i,n}$ according to (4) in the literature [10] and Pareto-dominated rules;
Update the external memory based on Pareto-dominated rules and Section IV.*B*);
end for
end while

V. Experiments

A. Algorithm Performance Evaluation

This algorithm uses convergence, distribution and operation time as evaluation parameters. Calculate the average distance between the optimal front obtained by experiment and the true optimal front according to (12), written as GD, which is used to evaluate convergence of algorithm.

$$GD = \frac{\sqrt{\sum_{i=1}^{n_s} d_i^2}}{n_s}, \quad (12)$$

where, n_s is the number of optimal solutions, d_i is Euclidean distance between the i^{th} point of the Pareto optimal front obtained by experiment and the nearest point of the true Pareto optimal front. As indicated by definition, the smaller of GD, the closer of optimal front obtained by experiment and the true optimal front, and the better convergence.

Distribution of algorithm is mainly measured by distribution scope SP. the smaller of SP, the more uniformly of optimal

solution distribute in objective space. The definition of SP is as:

$$SP = \sqrt{\frac{1}{n_s - 1} \sum_{i=1}^{n_s} (\bar{d} - d_i)^2}, \tag{13}$$

where, $d_i = \min_{j \neq i} \left\{ \sum_{n_i=1}^{n} |f_i^{n_i} - f_j^{n_i}| \right\}, \bar{d} = \frac{1}{n} \sum_{i=1}^{n_s} d_i$

B. Simulation Experiment

In this paper, we select the standard test functions ZDT1, ZDT3 and DTLZ2 to evaluate the performance of algorithms. These standard test functions have different characteristics, such as non-convexity, discontinuity, and so on,and they can be used to test searching ability of algorithms in different ways. ZDT1 and ZDT3 are the two-objective minimum problem, DTLZ2 is the three objective minimization problem.

C. Results

Set the population size $M = 100$, iterations $G = 100$, the size of external memory $C = 100$ and the grid is divided into 10. The parameters of MORDPSO-CD are set as $\alpha = 0.25, \beta = 1.95$, the parameters of MOQPSO-CD are set as $\alpha = 0.25$,, and $\omega = 0.1, c_1 = 2, c_2 = 2$ in PSO-In. Randomly select 30 experiments results, each experiment runs independently, three algorithms use the same initial particle swarm.

TABLE I
PERFORMANCE EVALUATION OF THREE ALGORITHMS BASED ON CROWDING DISTANCE

Index	*GD*			*SP*			*Time*		
	A	B	C	A	B	C	A	B	C
ZDT1	0.3661	0.3695	0.3665	0.0062	0.0065	0.0065	3.8554	9.4094	8.2304
ZDT3	0.1937	0.1872	0.1942	0.0110	0.0097	0.0097	3.5334	6.0494	6.1439
DTLZ2	0.3308	0.3327	0.3328	0.0506	0.0377	0.0687	6.6869	11.9155	8.6346

A, B, C express respectively MORDPSO-CD, MOQPSO-CD, MOPSO-in-CD.

Figs. 2-4 show the optimal front obtained by three algorithms based on crowding distance, the solid line in figure denotes the true Pareto optimal front of test function. Fig. 2 and Table I show that the MORDPSO-CD (Fig. 2 (a)) has better convergence and distribution than the other algorithms, running time is significantly less than other algorithms as well. MORDPSO-CD has the smallest GD, the best convergence properties, and the distribution of solution is in the middle level on ZDT1 (Fig. 2) and DTLZ2 (Fig. 4) problem. The convergence and distribution of MORDPSO-CD is second only to MOQPSO-CD on ZDT3 (Fig. 3) problem.

The optimal front achieved by MORDPSO-CD is better approximation to the true optimal front, and the other algorithms especially MOQPSO-CD does not get the optimal solution on the side and does not get some approximate solutions, as shown in Fig. 2. This phenomenon also appears in ZDT3, which has five sections discrete true optimal front. MORDPSO-CD gets five sections approximate solution, but the other algorithms do not find the solution in the same section (Figs. 3 (b) and (c)), this suggests that MORDPSO-CD has strong ability of searching the boundary solution. There is a large gap between some approximate solutions achieved by all algorithms on DTLZ2 problem and true optimal solution.

The results show that MORDPSO-CD has the best convergence and distribution on ZDT1 problem. Meanwhile, MORDPSO-CD has the best convergence but distribution of solution is in the middle level on DTLZ2 problem. However, the convergence and distribution rank only second to MOQPSO-CD.

VI. CONCLUSION

In this paper, we introduce RDPSO into multi-objective optimization problem and propose multi-objective Random Drift Particle Swarm Optimization algorithm based on crowding distance sorting (MORDPSO-CD). Besides, calculating the crowding distance, updating and maintaining the external memory, the choice of leader particle are discussed in detail. We adopt standard test functions and choose convergence, distribution and running time as performance indicator to compare MORDPSO-CD, MOQPSO-CD, MOPSO-In. The simulation on typical test functions indicates that this proposed algorithm has better global optimization and rapid convergence.

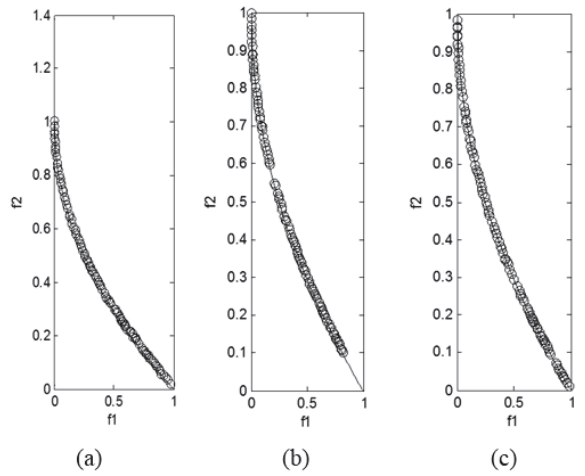

Fig. 2 The optimal front obtained by (a) MORDPSO, (b) MOQPSO, and (c) MOPSO-In based on crowding distance on ZDT1

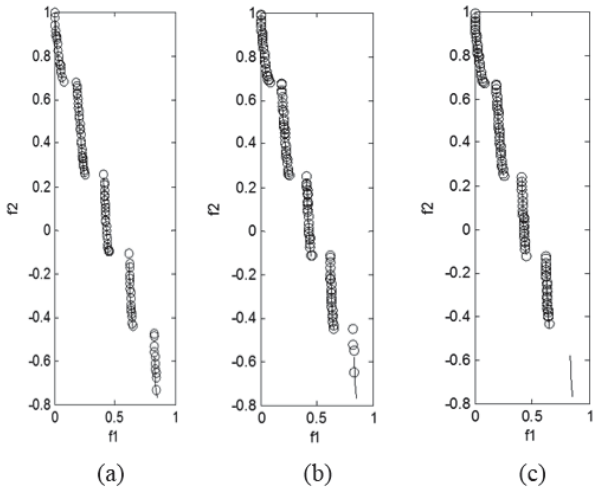

Fig. 3 The optimal front obtained by (a) MORDPSO, (b) MOQPSO, and (c) MOPSO-In based on crowding distance on ZDT3

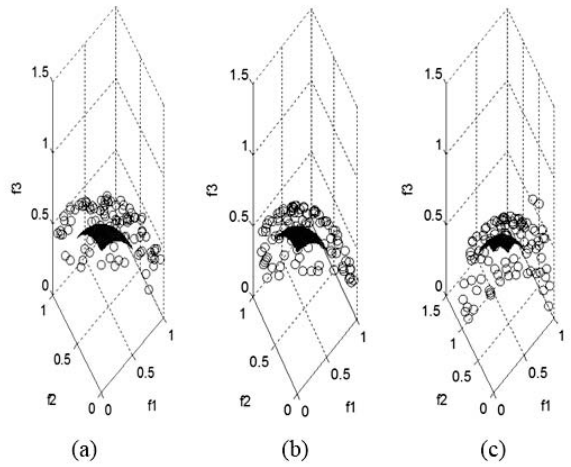

Fig. 4 The optimal front obtained by (a) MORDPSO, (b) MOQPSO, and (c) MOPSO-In based on crowding distance on DTLZ2

REFERENCES

[1] A. E. Smith, "Multi-objective optimization using evolutionary algorithms (Book Review)," Evolutionary Computation, IEEE Transactions on, vol. 6, pp. 526-526, 2002.

[2] C. A. Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," in Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, 2002, pp. 1051-1056.

[3] F. G. Guimaraes, E. F. Wanner, and R. H. C. Takahashi, "A quality metric for multi-objective optimization based on Hierarchical Clustering Techniques," in Evolutionary Computation, 2009. CEC '09. IEEE Congress on, 2009, pp. 3292-3299.K. Elissa, Title of paper if known, unpublished.

[4] M. Salazar-Lechuga and J. E. Rowe, "Particle swarm optimization and fitness sharing to solve multi-objective optimization problems," in Evolutionary Computation, 2005. The 2005 IEEE Congress on, 2005, pp. 1204-1211 Vol. 2.

[5] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," Evolutionary Computation, IEEE Transactions on, vol. 8, pp. 256-279, 2004.

[6] Y. J. J, Z. J. Z, R. C. fang, and e. al., "Multi-objective particle swarm optimization based on adaptive grid algorithm," J of System Simulation, pp. 5843-5847, 2008.

[7] Sun J, Lai C H, Wu X J. Particle swarm optimisation: classical and quantum perspectives (M). CRC Press, 2011.

[8] Raquel C R, Jr P C N. An effective use of crowdingdistance in multiobjective particle swarm optimization (C). Proc of the 2005 Workshops on Genetic and Evolutionary Computation. Washington: ACM Press, 2005: 257-264.

[9] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," Evolutionary Computation, vol. 8, pp. 173-195, 2000.

[10] J. Sun, X. Wu, V. Palade, W. Fang, and Y. Shi, "Random drift particle swarm optimization," arXiv preprint arXiv:1306.2863, 2013.

[11] S. Jun, V. Palade, W. Xiao-Jun, F. Wei, and W. Zhenyu, "Solving the Power Economic Dispatch Problem With Generator Constraints by Random Drift Particle Swarm Optimization," Industrial Informatics, IEEE Transactions on, vol. 10, pp. 222-232, 2014.

[12] L. Liqin, Z. Xueliang, X. Liming, and D. Juan, "A novel multi-objective particle swarm optimization based on dynamic crowding distance," in Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on, 2009, pp. 481-485.